CP1406 Week 8 Practical – Forms and JavaScript

For today's practical, you need to submit:

A single zip file containing

- **The Fitness folder**. The "Fitness" folder includes:
  - html files (e.g., index.html, template.html, contact.html, and aboutus.html) you have created for the site
  - a CSS file
  - a JavaScript file
  - the "images" folder containing images for the "Fitness" website.
- **The Form Folder** containing form.html, submitted.html and script.js.

Part 1 – Adding a Form, Fieldset, Legend, Labels, and Text Input Controls to the Contact Us Page

Let's create a form on the Contact Us page to collect information from prospective clients. The Forward Fitness Club wants to create a list of prospective clients and use it to market their services. Most of this client data can come from a form on the Contact Us page.

- Open **contact.html** in your editor.
- Above the closing **</main>** tag, add the following code (Figure 1):

```
    <div id="form">
        <h2>Complete the form below to begin your free trial.</h2>
        <form class="form-grid"> <!-- Start Form -->
            <fieldset>
                <legend>Customer Information</legend>
                <label for="fName">First Name:</label>
                <input type="text" name="fName" id="fName">

                <label for="lName">Last Name:</label>
                <input type="text" name="lName" id="lName">
            </fieldset>
        </form>
    </div>
</main>
```

- Save your changes and refresh contact.html in your browser.
  Note: "lName" is for "last name", so that's an l, not an I. Totally different, right?

Part 2 – Adding email and tel Input Controls to a Form

Add **email** and **tel** input controls to collect the customer's email address and telephone number respectively. Using these input controls instead of text controls makes it easier for visitors to enter the appropriate information and for the browser to automatically validate the inputs (in some situations).

- Above the closing **</fieldset>** tag, add the following code (Figure 1):

```
        <label for="email">Email:</label>
        <input type="email" name="email" id="email">

        <label for="phone">Phone: </label>
        <input type="tel" id="phone" name= "phone">
</fieldset>
```

- Save your changes and refresh contact.html in your browser.

Part 3 – Adding Checkbox Controls to a Form

Now let's add **checkbox** controls to collect information from customers about their primary interests in the fitness club. Using checkbox controls lets customers select more than one option (unlike radio buttons, which allow only a single option).

- Under the closing **</fieldset>** tag, add the following code (Figure 1):

```
<fieldset>
        <legend>Additional Information</legend>
        <p>I would like more information about:</p>

        <label for="grpfit"><input type="checkbox" name="interest" id="grpfit" value="Group
        Fitness">Group  Fitness</label>

        <label for="prtrain"><input type="checkbox" name="interest" id="prtrain"
        value="Personal Training">Personal Training</label>

        <label for="nutr"><input type="checkbox" name="interest" id="nutr"
        value="Nutrition">Nutrition</label>
</fieldset>
```

- Save your changes and refresh contact.html in your browser.

Part 4 – Adding a select Element to a Form

We will use a **select** element to learn how customers discovered the Forward Fitness Club.
Why? A select element lets users select from a list of options, restricting them to a valid option.

- Under the closing **</fieldset>** tag, add the following code (Figure 1):

```
<fieldset>
    <legend>Referral Source</legend>
    <label for="reference">How did you find us?</label>
    <select name="reference" id="reference">
        <option value="ad">Advertisement</option>
        <option value="friend">Friend</option>
        <option value="google">Google</option>
        <option value="social">Social Media</option>
        <option value="other">Other</option>
    </select>
</fieldset>
```

- Save your changes, refresh contact.html in your browser, and then tap or click the new select control to test that it works as expected.

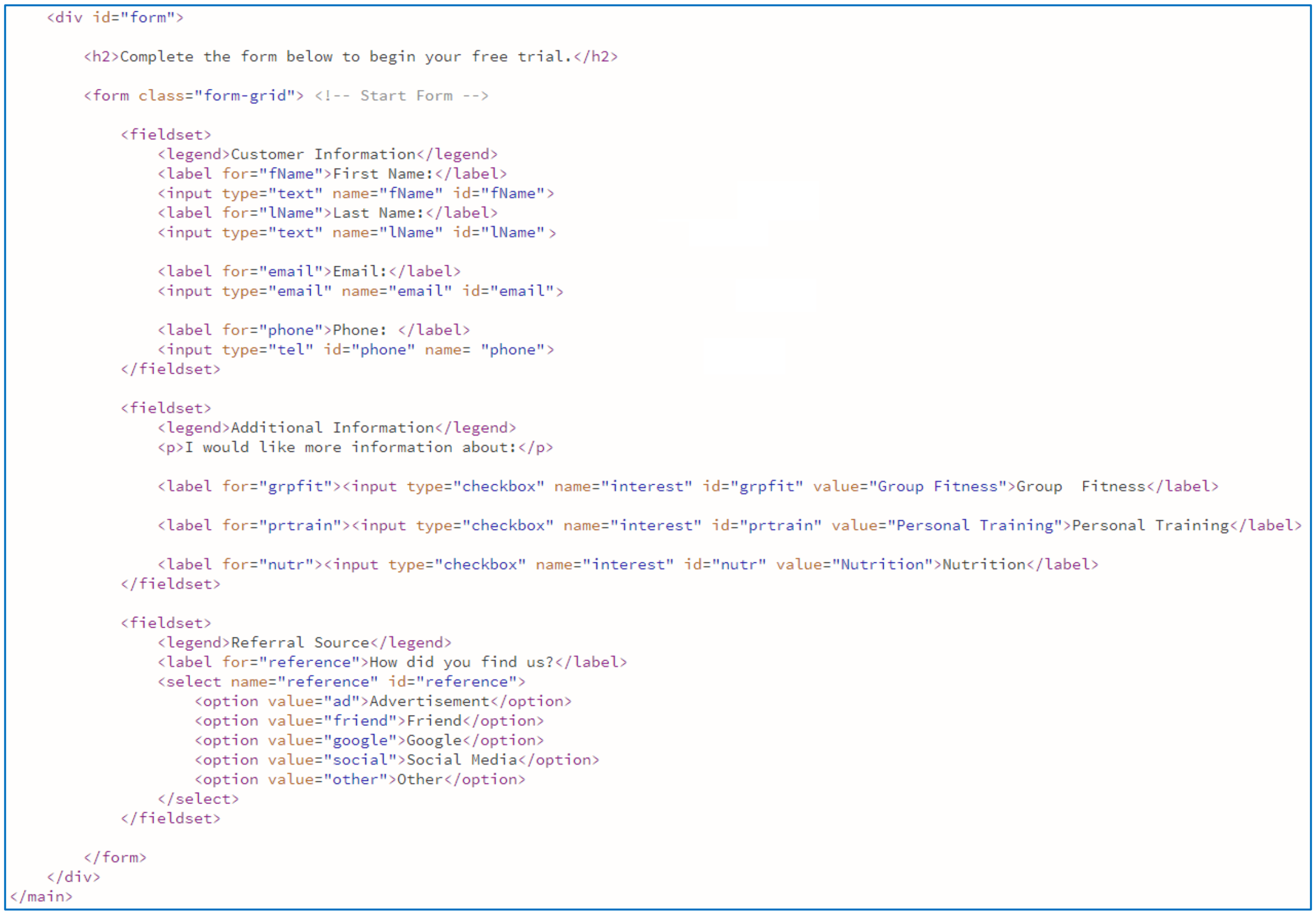
Part 5 – Adding a textarea Element to a Form

Add a **textarea** element to the form to provide an opportunity for customers to ask questions. Customers can enter more than one line of text in a textarea control.

- Between the closing **</select>** tag and the closing **</fieldset>** tag, add the following code (Figure 2):

```
    </select>

    <label for="questions"> Questions?</label>
    <textarea id="questions"  name="questions" rows="5" cols="35"></textarea>
</fieldset>
```

- Save your changes and refresh contact.html in your browser.



```
    <div id="form">

        <h2>Complete the form below to begin your free trial.</h2>

        <form class="form-grid"> <!-- Start Form -->

            <fieldset>
                <legend>Customer Information</legend>
                <label for="fName">First Name:</label>
                <input type="text" name="fName" id="fName">
                <label for="lName">Last Name:</label>
                <input type="text" name="lName" id="lName" >

                <label for="email">Email:</label>
                <input type="email" name="email" id="email">

                <label for="phone">Phone: </label>
                <input type="tel" id="phone" name= "phone">
            </fieldset>

            <fieldset>
                <legend>Additional Information</legend>
                <p>I would like more information about:</p>

                <label for="grpfit"><input type="checkbox" name="interest" id="grpfit" value="Group Fitness">Group  Fitness</label>

                <label for="prtrain"><input type="checkbox" name="interest" id="prtrain" value="Personal Training">Personal Training</label>

                <label for="nutr"><input type="checkbox" name="interest" id="nutr" value="Nutrition">Nutrition</label>
            </fieldset>

            <fieldset>
                <legend>Referral Source</legend>
                <label for="reference">How did you find us?</label>
                <select name="reference" id="reference">
                    <option value="ad">Advertisement</option>
                    <option value="friend">Friend</option>
                    <option value="google">Google</option>
                    <option value="social">Social Media</option>
                    <option value="other">Other</option>
                </select>
            </fieldset>

        </form>
    </div>
</main>
```

Figure 1

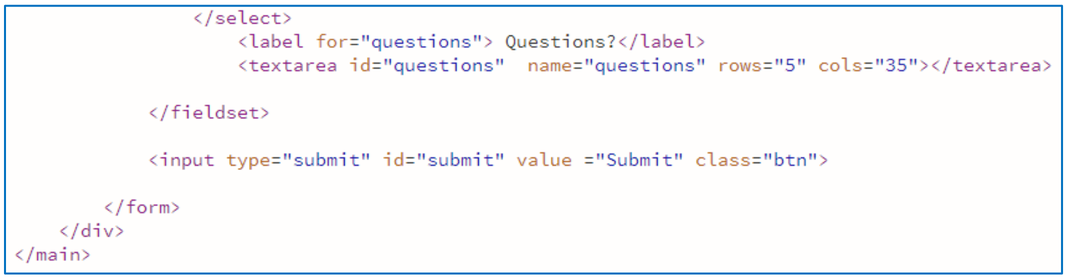
Part 6 – Adding a Submit Button to a Form

After adding text input controls and other form elements to a form, insert a **submit** control so that visitors can submit the form with their responses. Every form requires a submit control to send the information to a web server for processing if specified.

- Below the closing **</fieldset>** tag, add the following code (Figure 2):

```
        </fieldset>
        <input type="submit" id="submit" value ="Submit" class="btn">
    </form>
```

- Save your changes and refresh contact.html in your browser.



```
                    </select>
                        <label for="questions"> Questions?</label>
                        <textarea id="questions"  name="questions" rows="5" cols="35"></textarea>

                </fieldset>

                <input type="submit" id="submit" value ="Submit" class="btn">

            </form>
        </div>
</main>
```

Figure 2

Part 7 – Styling a Form for a Mobile Viewport

Now that the form is complete, you can style the form for a mobile viewport.
First, create a style rule for the form div to set some top margin, specify a background colour, and set some padding. Next, create a style rule to centre-align the heading 2 element within the form div. Create a style rule that sets the bottom margin for the fieldsets and controls.

- Download **styles.css** from Week 8 prac folder and save it to your **Fitness** folder. This file contains the css code for this week's prac work.
- Open **styles.css** to prepare to modify it.
- Above the **footer p** style rule, add the following code:

```
#form {
        margin-top: 2%;
        background-color: #f2f2f2;
        padding: 2%;
}

#form h2 {
        text-align: center;
}

/* Style rules for form elements */
fieldset, input,  select, textarea {
        margin-bottom: 2%;
}

fieldset legend {
        font-weight: bold;
        font-size: 1.25em;
  }

label {
        display: block;
        padding-top: 3%;
}

form #submit {
        margin: 0 auto;
        border: none;
        display: block;
        padding: 2%;
        background-color: #b3b3b3;
        font-size: 1em;
        border-radius: 10px;
}

/* Style rules for footer content */
footer p {
```

- Save your changes, refresh contact.html in your browser, and select the tablet viewport, and then scroll down if necessary to view the changes.

Part 8 – Styling a Form for a Tablet Viewport

Now that you have styled the form for a mobile viewport, you can also style the form for a tablet viewport.

- Within **@media screen and (min-width: 630px) { }**, under **#exercises dd** closing bracket (**}**), add the following code:

```
/* Tablet Viewport: Style rule for form element */
form {
        width: 70%;
        margin: 0 auto;
}
```

- Save your changes, refresh **contact.html** in your browser, and select the tablet viewport, and then scroll down if necessary to view the changes.

Part 9 – Styling a Form for a Desktop Viewport

Now that you have styled the form for mobile and tablet viewports, you can style the form for a desktop viewport.

- Locate **@media screen and (min-width: 769px) {**, change the min-width to 1015px.
- Within **@media screen and (min-width: 1015px) { }**, under the closing bracket (**}**) of the **#exercises** style rule, add the following code:

```
#exercises {
        clear: left;
}

/* Desktop Viewport: Style rules for form elements */
form {
        width: auto;
}
```

```
.form-grid {
        display: grid;
        grid-template-columns: auto auto auto;
        gap: 20px;
}

.btn  {
        grid-column: 1 / span 3;
}
```

- Save your changes, refresh **contact.html** in your browser, and display the webpage in a desktop viewport.

JavaScript

JavaScript is scripting language used to provide various types of functionality to webpages, such as the ability to interact with the user. In a web environment, web developers use a scripting language to control webpages. This includes providing user feedback, validating form data, and creating dynamic content based on user actions. JavaScript is a client-side scripting language, which means that the browser (i.e., the client) interprets and renders the JavaScript. Most modern websites use some form of JavaScript within their website. JavaScript provides additional interactivity between the webpage and the user.

Part 10 – Using JavaScript to Validate a Form

To practise using JavaScript to validate a form, we will create a simple html form and use JavaScript to check for an entry to the fields. We will create 3 files: an HTML file with the form, an HTML file to notify the users that their form has been submitted successfully, and a JavaScript file with the validation function. See https://www.w3schools.com/js/js_validation.asp for more information.

- Create a folder called "**Form**", create, and save an html file called "**form.html**" in the "**Form**" Folder.
- Add the following code to the **form.html**:

```
<!doctype html>
<html>
<head>
        <meta charset="utf-8">
        <title>Simple Form</title>
        <script src="script.js"></script> <!-- Link JavaScript to HTML -->
</head>

<body>
```

```
        <form name="myForm" action="submitted.html" onsubmit="return
        validateForm()">
                Name: <input type="text" name="name">
                Email: <input type="text" name="email">
                <input type="submit" value="Submit">
        </form>

</body>
</html>
```

- Save your changes, open the file in a web browser, click "**Submit**" button and see the effects.
- Create a JavaScript file called "**script.js**" and save the file to the "**Form**" folder. The **script.js** file contains the following code:

```
function validateForm() {
        var name = document.forms["myForm"]["name"].value;
        var email = document.forms["myForm"]["email"].value;
        if (name == "") {
                alert("Name must be filled out");
                return false;
        }
        if (email == "") {
                alert("Email must be filled out");
                return false;
        }
}
```

- Save your changes.

- Create an html file called "**submitted.html**" and save it in the "**Form**" folder. The **submitted.html** file contains the following code:

```
<!doctype html>
<html>
<head>
        <meta charset="utf-8">
        <title>Your Form Has Been Submitted</title>
</head>

<body>
        <h1>Your Form Has Been Submitted</h1>
</body>
</html>
```

- Save your changes, open **form.html** in a web browser, click "**Submit**" button and see the effects.

Your Turn: Try extending what you have learnt just now to validate the form you have created from the steps in Part 1 – 6. See https://www.w3schools.com/js/js_validation.asp for more information.

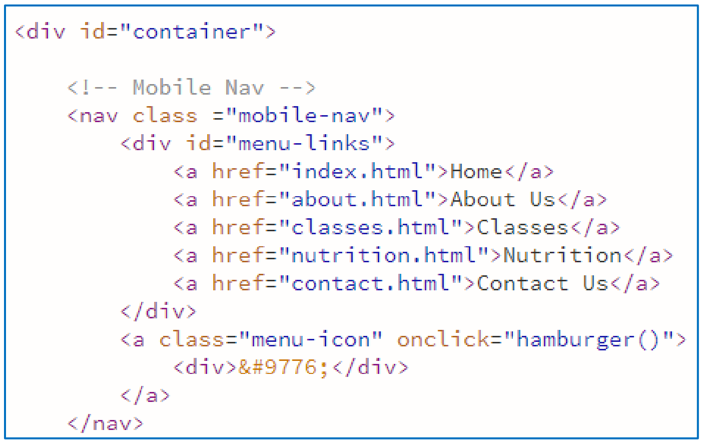
Part 11 – Creating a New Nav Element for a Mobile Viewport

You will add a new nav element to your HTML webpages. This new nav element will be used to display a hamburger menu for a mobile viewport.

- In **index.html**, below the **<div id="container">**, add the following code to add a new nav element, div elements, and anchor elements (Figure 3):

```
<body>
        <div id="container">

                <!-- Mobile Nav -->
                <nav class="mobile-nav">
                        <div id="menu-links">
                                <a href="index.html">Home</a>
                                <a href="about.html">About Us</a>
                                <a href="classes.html">Classes</a>
                                <a href="nutrition.html">Nutrition</a>
                                <a href="contact.html">Contact Us</a>
                        </div>
                        <a class="menu-icon">
                                <div>&#9776;</div>
                        </a>
                </nav>
```

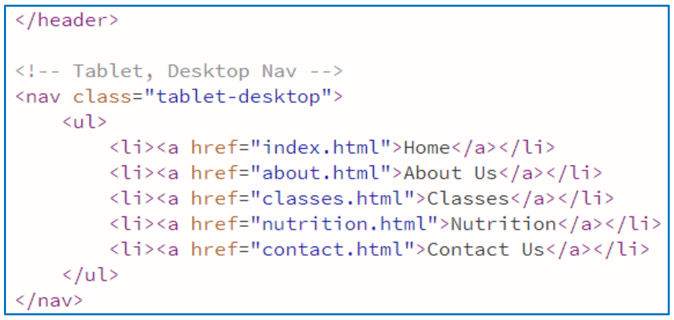
- **Note:** The purpose of **<a class="menu-icon">** anchor element is that this anchor element will be used to toggle the hamburger menu on and off. The 9776 entity code is the hamburger (three lines) icon.
- Save your changes then open the **index.html** file in Google Chrome's device mode, and then click the Mobile bar to view the page in a mobile viewport.



```
<div id="container">

    <!-- Mobile Nav -->
    <nav class ="mobile-nav">
        <div id="menu-links">
            <a href="index.html">Home</a>
            <a href="about.html">About Us</a>
            <a href="classes.html">Classes</a>
            <a href="nutrition.html">Nutrition</a>
            <a href="contact.html">Contact Us</a>
        </div>
        <a class="menu-icon" onclick="hamburger()">
            <div>&#9776;</div>
        </a>
    </nav>
```

Figure 3

- Under the closing **</header>** tag, replace the existing comment with the text Tablet, Desktop Nav to modify the comment (Figure 2).
- Add a class attribute class="tablet-desktop" to the nav tag (Figure 4).



```
</header>

<!-- Tablet, Desktop Nav -->
<nav class="tablet-desktop">
    <ul>
        <li><a href="index.html">Home</a></li>
        <li><a href="about.html">About Us</a></li>
        <li><a href="classes.html">Classes</a></li>
        <li><a href="nutrition.html">Nutrition</a></li>
        <li><a href="contact.html">Contact Us</a></li>
    </ul>
</nav>
```

Figure 4

- Repeat all the Part 1 steps to **about.html**, **contact.html**, and **template.html** files.

Part 12 – Styling the New Nav Element for a Mobile Viewport

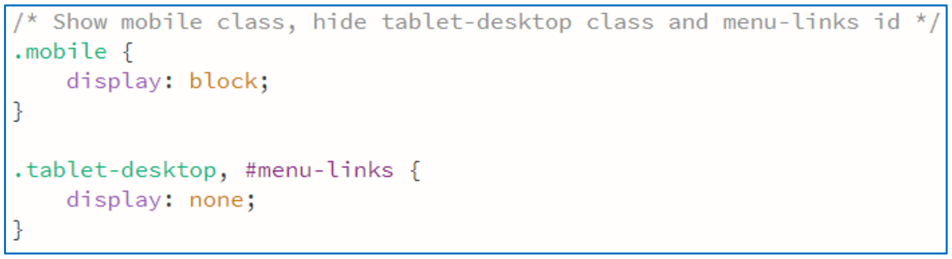
Now that you have added the hamburger icon, you will style it for a mobile viewport to show the hamburger icon and hide the new nav links. You will use JavaScript in future steps to toggle the links to be displayed. Create a style rule for the menu-links id to format the appearance of the navigation links when displayed. Create a style rule to format the appearance and the position of the hamburger icon.

- Open the **styles.css** file in your text editor to prepare to edit the file.
- Locate and replace the comment text "**navigation area**" with hamburger menu.
- Above **nav {** style rule, add the following code to add a new selector **.mobile-nav a**, which targets anchor elements in the mobile-nav class:

```
/* Style rules for hamburger menu */
.mobile-nav a {
        color: #fff;
        font-family: 'Francois One',  sans-serif;
        text-align: center;
        font-size: 2em;
        text-decoration: none;
        padding: 3%;
        display: block;
}

nav {
        padding: 1%;
        margin-bottom: 1%;
}
```

- Save your changes and open the file in Google Chrome's device mode to see the effects.
- Your turn - you will now create **.mobile-nav a.menu-icon** selector which targets the hamburger icon within a div element with the class attribute menu-icon. This div is nested within an anchor element, which is contained in the mobile-nav class.
    - Below the closing bracket (**{**) of **.mobile-nav a** style rule, create a selector .mobile-nav a.menu-icon.
    - This selector includes the following property values:
        - display as block
        - absolute position
        - right = 0
        - top = 0
    - Save your changes and refresh the index.html file in Google Chrome's device mode.

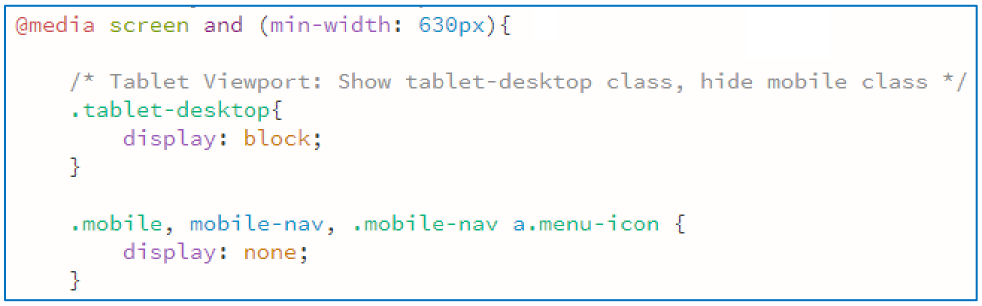
- You will add the **#menu-links** selector to the style rule. This style rule sets the display to none. When the page opens, you do not want the links displayed. Instead, you will use JavaScript to display the navigation links when the user taps the hamburger icon. The following steps add the **#menu-links** selector to the style rule.
    - Add the text and menu-links id to the end of the "/* Show mobile class, hide tablet-desktop class */" comment to modify the comment (Figure 5).
    - After **.tablet-desktop**, type , #menu-links to add a new selector to this style rule (Figure 5).



```
/* Show mobile class, hide tablet-desktop class and menu-links id */
.mobile {
    display: block;
}

.tablet-desktop, #menu-links {
    display: none;
}
```

Figure 5

- Locate .mobile style rule within the @media screen and (min-width:630px) {}, add new selectors: , mobile-nav, .mobile-nav a.menu-icon to mobile style rule (Figure 6).
- Save your changes and refresh the index.html file in Google Chrome's device mode.



```
@media screen and (min-width: 630px){

    /* Tablet Viewport: Show tablet-desktop class, hide mobile class */
    .tablet-desktop{
        display: block;
    }

    .mobile, mobile-nav, .mobile-nav a.menu-icon {
        display: none;
    }
```

Figure 6

Part 13 – Modifying Previous Navigation Style Rules for a Mobile Viewport

You have finished styling the hamburger menu and navigation links but need to do some housekeeping regarding the previous style rules created for the navigation links for the mobile viewport. As a developer, you should always keep your code clean and remove unnecessary code. Move the mobile style rules for the nav, nav ul, and nav li a to the tablet media query. Move two declarations from the nav li for mobile to the nav li style rule for tablet. Remove the nav li and nav li:first-child style rules for mobile. The following steps modify style rules for mobile and tablet viewports in the styles.css file.

- Locate and move (cut and paste) the following code to the line below the comment: Tablet Viewport: Style rules for nav area , located within the "Tablet media query" (Figure 7):

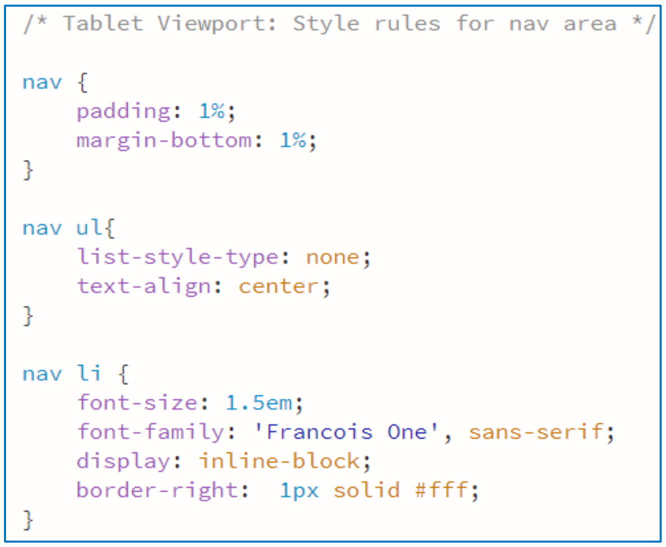
```
nav {
        padding: 1%;
        margin-bottom: 1%;
}

nav ul{
        list-style-type: none;
        text-align: center;
}
```

- Locate the following **nav li** style rules:

  ```
  nav li {
          font-size: 1.5em;
          font-family: 'Francois One', sans-serif;
          border-top: 1px solid #fff;
  }
  ```

- Cut and paste the following code to the **nav li** style rule , located within the "**Tablet media query**" (Figure 7):

  ```
  font-size: 1.5em;
  font-family: 'Francois One', sans-serif;
  ```



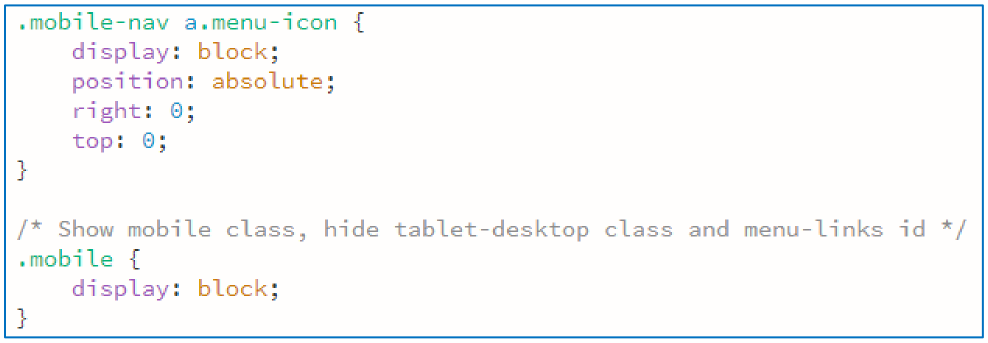
```
/* Tablet Viewport: Style rules for nav area */

nav {
    padding: 1%;
    margin-bottom: 1%;
}

nav ul{
    list-style-type: none;
    text-align: center;
}

nav li {
    font-size: 1.5em;
    font-family: 'Francois One', sans-serif;
    display: inline-block;
    border-right:  1px solid #fff;
}
```

Figure 7

- Scroll back up to the mobile **nav li** style rules and delete the **nav li** and the **nav li: first-child** style rules.
- Remove the **padding** declaration for **nav li a** style rule.
- Select the following code and **cut** (don't delete) these three declarations out of the **nav li a** style rule:

  ```
  display: block;
  color: #fff;
  text-decoration: none;
  ```

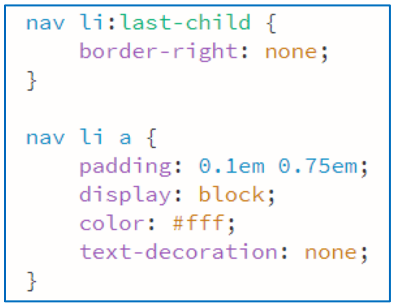
- Remove the **nav li a** style rule (see Figure 8)



```
.mobile-nav a.menu-icon {
    display: block;
    position: absolute;
    right: 0;
    top: 0;
}

/* Show mobile class, hide tablet-desktop class and menu-links id */
.mobile {
    display: block;
}
```

Figure 8

- Scroll down to the **tablet media query** and locate the **nav li a** style rule, paste the three declarations within the **nav li a** style rule (Figure 9).



```
nav li:last-child {
    border-right: none;
}

nav li a {
    padding: 0.1em 0.75em;
    display: block;
    color: #fff;
    text-decoration: none;
}
```

Figure 9

- Save your changes and refresh the **index.html file** in Google Chrome's device mode and select the mobile viewport to view the hamburger menu icon.

Part 14 – Creating a JavaScript File and the hamburger() Function

Now let's write the JavaScript to make our hamburger menu work. The HTML files all need to know the location of the JavaScript file to use it.

- Open your text editor, create a new file, **script.js**, in your fitness folder.
- In the **index.html**, **about.html**, **contact.html**, and **template.html**, above the closing **</body>** tag, add the following code to insert a script element:

  <script src="script.js"></script>

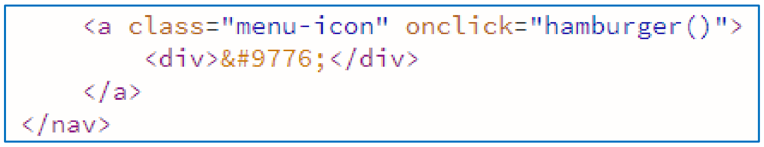
The hamburger menu is almost ready. You now need to create a JavaScript function to make the hamburger menu functional. JavaScript programming is needed to display and hide the navigation links for a mobile viewport. Use the external JavaScript file, **script.js**, to write a function that displays the navigation links when the hamburger icon is selected.

- Add the following code to script.js:

```
// Hamburger menu function
function hamburger() {
        var menu = document.getElementById("menu-links");
        var logo = document.getElementById("ffc-logo");
        if (menu.style.display === "block" && logo.style.display === "none") {
                menu.style.display = "none";
                logo.style.display = "block";
        } else {
                menu.style.display = "block";
                logo.style.display = "none";
        }
}
```

- In **index.html**, add the **onclick** event handler to the **menu-icon** class anchor element using the following code (Figure 10):

```
onclick="hamburger()"
```



```
    <a class="menu-icon" onclick="hamburger()">
        <div>&#9776;</div>
    </a>
</nav>
```

Figure 10

- Open the **index.html file** in Google Chrome's device mode to view the page in a mobile viewport.
- Tap or click the hamburger icon to display the navigation links.

Deliverables:

A single zip file containing

- **The Fitness folder**. The "Fitness" folder includes:
  - html files (e.g., index.html, template.html, contact.html, and aboutus.html) you have created for the site
  - a CSS file
  - a JavaScript file
  - the "images" folder containing images for the "Fitness" website.
- **The Form Folder** containing form.html, submitted.html and script.js.